House Price Prediction using Machine Learning

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

by

Rojalin Behera (Roll No. 05)

Siddhi Katkar (Roll No. 15)

Ziyaad Sayyad (Roll No. 42)

Nasir Shaikh (Roll No. 49)

Under the guidance of

Prof. Mubashir Khan

Department of Computer Engineering,

Theem College Of Engineering

Village Betegaon, Boisar Chilhar Road, Boisar (E), Palghar

2023-2024

House Price Prediction using Machine Learning

Submitted in partial fulfilment of the requirements of the degree of

Bachelor of Engineering

by

Rojalin Behera (Roll No. 05)

Siddhi Katkar (Roll No. 15)

Ziyaad Sayyad (Roll No. 42)

Nasir Shaikh (Roll No. 49)

Under the guidance of

Prof. Mubashir Khan

Department of Computer Engineering,

Theem College Of Engineering

Village Betegaon, Boisar Chilhar Road, Boisar (E), Palghar

University Of Mumbai

2023-2024

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principals of academics honestly and integrity have not misrepresented or fabricated or falsified any idea/data/fact/sources in my submission. We understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the source which has thus not been properly cited or from whom proper permission has not been taken when needed.

Rojalin Behera 05

Siddhi Katkar 15

Ziyaad Sayaad 42

Nasir Shaikh 49

Date:

Project Report Approval for Bachelor Of Engineering

The project report entitled **House Price Prediction using Machine Learning** by Rojalin Behera, Siddhi Katkar, Ziyaad Sayyad, Nasir Shaikh is approved for the degree of Bachelor of Engineering in Computer Engineering.

Date:

Place:

Examiners:

Name & Sign of External Examiner

Name & Sign of Internal Examiner

CERTIFICATE

This is to certify that the project entitled **"House Price Prediction using Machine Learning"** is a bonafide work of **"Rojalin Behera(05), Siddhi Katkar(15), Ziyaad Sayyad(42), Nasir Shaikh(49)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering" in "Computer Engineering"** has been carried out under my supervision at the department of Computer Engineering of Theem College of Engineering, Boisar. The work is comprehensive, complete and fit for evalautaion.

Prof. Mubashir Khan
Project Guide

Prof. Monika Pathare
Project Coordinator

Prof. Mubashir Khan
HOD

Dr.Riyazoddin Siddique
Principal

Acknowledgement

First and foremost, we thank God Almighty for blessing us immensely and empowering us at times of difficulty like a beacon of light. Without His divine intervention we wouldn't have accomplished this project without any hindrance.

We are also grateful to the Management of Theem College of Engineering for their kind support. Moreover, we thank our beloved Principal **Dr.Riyazoddin Siddiqui**, our Director, **Dr.N.K. Rana** for their constant encouragement and valuable advice throughout the course.

We are profoundly indebted to **Prof. Mubashir Khan**, Head of the Department of Computer Engineering and **Prof. Monika Pathare,** Project Coordinator for helping us technically and giving valuable advice and suggestions from time to time. They are always our source of inspiration. Also, we would like to take this opportunity to express our profound thanks to our guide **Prof.Mubashir Khan ,** Assistant Professor, Computer Engineering for his/her valuable advice and whole hearted cooperation without which this project would not have seen the light of day.

We express our sincere gratitude to all Teaching/Non-Teaching staff members of Computer Engineering department for their co-operation and support during this project.

Rojalin Behera (05)
Siddhi Katkar (15)
Ziyaad Sayyad(42)
Nasir Shaikh (49)

ABSTRACT

House price prediction is a fundamental task in real estate economics, crucial for various stakeholders including buyers, sellers, investors, and policymakers. This abstract presents a comprehensive overview of model-based approaches employed for house price prediction.

We explore the application of various machine learning algorithms such as linear regression, decision trees, random forests in predicting house prices. Leveraging datasets comprising features like property attributes, location, amenities, and neighborhood characteristics, we evaluate the performance of these models using metrics like mean absolute error, mean squared error.

Additionally, we investigate the impact of feature engineering, data preprocessing techniques, and model hyper parameters on prediction accuracy. Furthermore, we assess the generalization capabilities of the models across different geographical regions and time periods.

Our analysis aims to provide insights into the strengths and limitations of model-based approaches for house price prediction, facilitating informed decision-making in the real estate market and contributing to the advancement of predictive modeling techniques in this domain.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Chapter 1

INTRODUCTION

In today's dynamic real estate market, accurate prediction of house prices is essential for various stakeholders, including buyers, sellers, investors, and policymakers. Traditional methods of estimating property values often fall short in capturing the complexities and nuances inherent in the housing market. However, with the advent of advanced modeling techniques, particularly machine learning algorithms, the ability to predict house prices with greater precision has significantly improved.

This introduction sets the stage for exploring model-based house price prediction, where we delve into the application of sophisticated analytical methods to forecast property values. By harnessing vast datasets containing diverse property attributes, market indicators, and socio-economic factors, these models offer insights into the factors driving housing prices and enable stakeholders to make informed decisions.

We embark on a journey to explore the landscape of model-based house price prediction. We examine various machine learning algorithms, such as linear regression, decision trees, random forests evaluating their efficacy in forecasting house prices. Additionally, we delve into the methodologies employed for data preprocessing, feature engineering, and model optimization to enhance predictive accuracy.

Furthermore, we investigate the challenges and opportunities associated with model-based house price prediction, including model interpretability, data quality issues, and the generalization of models across different geographical regions and time periods. Through a comprehensive analysis, we aim to provide valuable insights into the strengths and limitations of these approaches and their implications for real estate stakeholders.

Overall, this exploration into model-based house price prediction underscores the importance of leveraging advanced analytical tools to navigate the complexities of the real estate market. By harnessing the power of data and analytics, we strive to empower stakeholders with actionable insights and facilitate more informed decision-making in the pursuit of understanding and forecasting housing prices.

Chapter 2

LITERATURE REVIEW

2.1 House price prediction via improved Machine Learning Techniques

Authors: Quang Troung mint Nguyen,Hy dang ,bo Mei

Summary:House price index is used to measure price of residential house in many countries like US Federal Housing Finance Agency HPI,UK National Statistics HPI, Singapore HPI,etc. Methodologies used are Data pre processing,Model Selection,Random Forest,XG BOOST,Hybrid Regression,Stacked Generalisation.

2.2 House Price Prediction using Regression Techniques

Authors: Master Amit Kumar, Karun Kumar Singh

Summary:The log transformation can be used to make highly skewed distribution less skewed.This is valuable for both ,for making pattern in the data more interpretable helping to meet the assumption for inferential statistics. Libraries used: 1.Pandas 2.Numpy 3.Matplotlib 4.Seaborn 5.Scikit Learn 6.XG BOOST

2.3 House Price Prediction using machine learning

Authors: Prof.Balaraman Ravindran(IIT Madras)

Summary: Design approach Linear regression allows us to summarize and study the relationship between two continuous quantitative variables.This model with LASSO regression provides an accuracy of 65 percent.

2.4 House Price forecasting using machine learning

Authors: Nihar Bhagwat , Ankit Mohokar, Shreyansh Mane

Summary:The target feature in this purpose model is the price of the real estate property and independent feature are number of bathroom,number of bedrooms,carpet area,edge of property. The system provides 89 percent accuracy while predicting the prices for the flats.

Chapter 3

LIMITATIONS OF EXISTING SYSTEM OR RESEARCH GAP

1. Limited amount of work has been focused of the house price prediction model.
2. Most of the past research considered the housing market problem as a classification problem to develop a classification model instead of regression model.
3. Older models have been not so precise as a accuracy level was the least compared to newer models.
4. This paper did not cover all the regression algorithm,instead they are focused on the chosen algorithm, staring for the basic regression techniques to the advanced once .

Chapter 4

PROBLEM STATEMENT AND OBJECTIVE

4.1 Problem Statement

Developing accurate models to forecast residential property prices is challenging due to factors such as data availability, model complexity, and ethical considerations. Key challenges include acquiring high-quality data, selecting appropriate modeling techniques, identifying relevant features, ensuring model generalization, providing transparent predictions, and addressing ethical concerns. Efficiently addressing these challenges is crucial for building reliable predictive models that facilitate informed decision-making in the real estate market.

- Informed Decision-Making: Buyers, sellers, investors, and policymakers rely on house price predictions to make informed decisions. Buyers use predictions to assess affordability and negotiate prices, while sellers utilize them to determine listing prices.

- Market Analysis: House price predictions provide valuable insights into market trends and dynamics. Analyzing predicted prices over time helps stakeholders understand market fluctuations,identify emerging trends, and anticipate future market conditions.

- Risk Mitigation: Predicting house prices assists in managing risks associated with real estate transactions. By forecasting potential price changes, stakeholders can assess risks and develop strategies to mitigate them, such as diversifying investment portfolios or timing property purchases/sales.

- Financial Planning: House price predictions aid individuals and organizations in financial planning. Home buyers use predictions to estimate mortgage affordability, while lenders and financial institutions use them to assess loan risks and determine lending terms.

Chapter 5

PROPOSED SYSTEM

5.1 System Architecture 1

This home price prediction system employs a three-tiered architecture. The frontend, built with Flask and web technologies, collects data and interacts with users. The backend, powered by Python, preprocesses data and trains the price prediction model using SVM and Random Forest algorithms. The system offers high accuracy, scalability, accessibility, and customization, making it a robust price prediction solution.

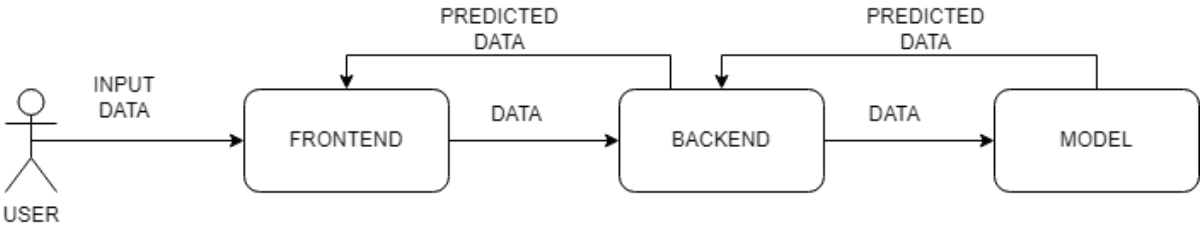

Figure 5.1.0.1: System Architecture 1

The system architecture consists of:

1. **Frontend:** The frontend of the system is responsible for collecting data from the network or system being monitored and displaying the results to users. The frontend is implemented using Flask, a lightweight web framework for Python. CSS is used to create the user interface of the model.

2. **Backend:** The backend of the system is responsible for preprocessing the collected data, training the Price detection model, deploying the model, and performing price detection. The backend is implemented using Python. Two machine learning algorithms, SVM and Random Forest, are used to train the intrusion detection model. Datasets are used to train and evaluate the model.

3. **Evaluation and Analysis:** This module is responsible for evaluating the performance of the trained house price prediction model on the two datasets and displaying the results to users in the frontend. The Evaluation and Analysis Module

can be implemented using a variety of tools and technologies, such as Python libraries such as scikit-learn and pandas.

1. **Data collection:** The frontend collects data from the network or system being monitored. This data can be in the form of sq. feet,location,no. of bedrooms,no. of bathrooms,RERA Scheme.

2. **Data preprocessing:** The backend preprocesses the collected data to extract features and prepare it for machine learning training.

3. **Model training:** The backend trains the intrusion detection model using the preprocessed data. Two machine learning algorithms, SVM and Random Forest, are used in the system.

4. **Model evaluation and analysis:** The backend evaluates the performance of the trained price detection model on the two datasets and displays the results to users in the frontend.

.

The following are some of the benefits of the proposed system architecture:

1. **Accuracy:** The system uses two machine learning algorithms, SVM and Random Forest, which are known for their high accuracy in house price detection.

2. **Scalability:** The system can be scaled to meet the needs of the system by deploying the house price detection model on the cloud.

3. **Flexibility:** The system can be customized to meet the specific needs of the organization.

4. **Evaluation and Analysis:** The system includes an Evaluation and Analysis Module that evaluates the performance of the trained price detection model and displays the results to users in the frontend.

5.2 System Architecture 2

This architecture is based on the idea of using a large and diverse dataset of labeled prices to train a machine learning model to detect prices.

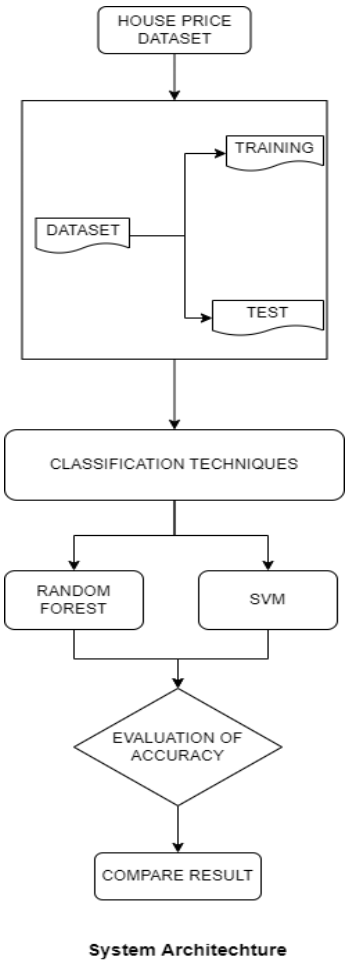

Figure 5.2.0.1: System Architecture 2

The system architecture works as follows:

1. **Data collection:** Data collection for a house price prediction model involves gathering information from various sources pertinent to real estate. These sources include property listings, real estate agencies, government databases, and public records. Property listings provide details on house features such as size, number of bedrooms and bathrooms, amenities, and overall condition. Real estate agencies offer insights into market trends, recent sales data, and comparable properties in the area. Government databases and public records provide information on property taxes, zoning regulations, and historical data on housing prices.

   The data collected is diverse in nature and may include structured data from property listings and sales records, as well as unstructured data from photographs

and property descriptions. Structured data is organized in a tabular format and can be easily analyzed, while unstructured data requires techniques such as natural language processing (NLP) and image processing to extract relevant information. By collecting and integrating data from multiple sources in different formats, a comprehensive dataset is created for training and evaluating the house price prediction model.

2. **Data preprocessing and feature engineering:** The collected data is preprocessed and engineered to extract features that are relevant for intrusion detection. This includes cleaning the data, removing noise, and converting the data into a format that is compatible with the machine learning algorithms. Feature engineering is the process of creating new features that are more informative for intrusion detection. This can be done by combining different features, creating new features based on existing features, or transforming features into a different format.

3. **Model training:** The machine learning models are trained on the engineered dataset. Two machine learning algorithms, SVM and Random Forest, are used in the system. SVM is a supervised learning algorithm that can be used for both classification and regression tasks. Random Forest is an ensemble learning algorithm that combines the predictions of multiple decision trees to produce a more accurate prediction.

4. **Model evaluation:** The trained models are evaluated on a held-out test set to assess their performance. This helps to identify any areas where the models need improvement. The evaluation metrics that are used depend on the specific machine learning algorithms that are used. For example, common evaluation metrics for classification tasks include accuracy, precision, recall, and F1 score.

5. **Model deployment:** The trained models are deployed to production. This can be done by deploying them to a cloud platform or to on-premises servers. The specific deployment method that is used depends on the specific requirements of the organization.

    (a) **House price detection:** The deployed models are used to detect prices in real time. The models analyze given data and generate price when they detect the requirements of the user.

6. **Evaluation and analysis:** It involves checking how accurate it is in predicting actual house prices. This is done by comparing the predicted prices with the

real prices using different measures like mean error and root mean squared error. Techniques like cross-validation help ensure the model works well with new data. By evaluating and analyzing the model's performance, we can understand its strengths and weaknesses and improve it over time for better predictions in the real estate market. The results of the evaluation and analysis are displayed to users in the frontend. This allows users to monitor the performance of the models over time and to make adjustments as needed.

5.3 Model Training

The SVM and Random Forest algorithms will be implemented to train the IDS models. SVM is a supervised learning algorithm that separates data points into different classes using hyperplanes in high-dimensional space. Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions. The training process involves splitting the datasets into training and testing sets. The models will be trained on the training set and evaluated on the testing set to measure their performance. Various metrics, such as accuracy, precision, recall, and F1-score, will be used to assess the models' effectiveness in detecting price.

5.3.1 Random Forest Algorithm:

Random Forest is a popular machine learning algorithm that is used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to make predictions. The algorithm gets its name from the fact that it creates a "forest" of decision trees, where each tree is built using a random subset of the training data. The key components of Random Forest are:

1. **Bootstrapping:** Random Forest employs bootstrapping, which is a sampling technique where multiple subsets of the original training data are created by randomly selecting samples with replacement. This means that some samples may appear multiple times in a subset, while others may not appear at all. Bootstrapping helps introduce diversity in the training data for each decision tree.

2. **Random Feature Selection:** In addition to using random subsets of the training data, Random Forest also uses random feature selection. At each node of a decision tree, only a subset of features is considered for splitting. This further enhances the diversity among the trees and prevents any single feature from dominating the decision-making process.

3. **Building Decision Trees:** Each decision tree in the Random Forest is built using a recursive process called recursive partitioning. The goal is to split the data at

each node based on a selected feature and its corresponding threshold value. The splitting criterion, such as Gini impurity or information gain, is used to determine the best feature and threshold for each split. The process continues until a stopping criterion is met, such as reaching a maximum depth or minimum number of samples in a leaf node.

4. **Voting and Aggregation:** Once all the decision trees are built, the Random Forest algorithm combines their predictions to make the final prediction. For classification tasks, the most common class predicted by the individual trees is selected as the final prediction. For regression tasks, the average or median of the predicted values is taken as the final prediction. This voting and aggregation process helps reduce the variance and improve the overall accuracy of the model.

Algorithm:

1. Initialize the number of trees (d) and the maximum depth of the tree (d)
2. For each tree:
   (a) Select a random subset of features (X) from the training data
   (b) Use a random subset of the training data (X) to train each decision tree
   (c) Use the remaining features (X) to train the next decision tree
   (d) Repeat steps b and c until all trees are trained
3. For each tree:
   (a) Compute the prediction for the test data (y) using the trained decision tree
   (b) Compute the error (e) between the predicted output (y) and the actual output (y')
4. Combine the predictions of all trees to produce the final output (y_rand_forest)

Implementation in Scikit-learn:
For each decision tree, Scikit-learn calculates a nodes importance using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \tag{5.1}$$

where,

- ni_j = the importance of node j
- w_j = weighted number of samples reaching node j
- C_j = the impurity value of node j
- $left_j$ = child node from left split on node j
- $right_j$ = child node from right split on node j

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\sum_{j:\text{node } j \text{ splits on feature } i} n_j}{\sum_{k \in \text{ all nodes}} n_k} \tag{5.2}$$

where,

- fi_i = the importance of feature i
- ni_j = the importance of node j

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$normfi_i = \frac{fi_i}{\sum_{j \in \text{ all features}} fi_j} \tag{5.3}$$

The final feature importance, at the Random Forest level, is it's average over all the trees. The sum of the feature's importance value on each trees is calculated and divided by the total number of trees:

$$RFfi_i = \frac{\sum_{j \in \text{ all trees } j} normfi_{ij}}{T} \tag{5.4}$$

where,

- $RFfi_i$ = the importance of feature i calculated from all trees in the Random Forest model
- $normfi_{ij}$ = the normalized feature importance for i in tree j
- T = total number of trees

5.3.2 Support Vector Machine Algorithm:

Support Vector Machines (SVM) is a powerful machine learning algorithm that is widely used for classification and regression tasks. It is based on the principles of statistical learning theory and aims to find an optimal hyperplane that separates different classes or predicts continuous values. In this explanation, we will delve into the mathematical terms and concepts behind SVM. Let's start by considering a binary classification problem where we have a set of labeled training data points. Each data point is represented by a feature vector x and belongs to one of two classes, either positive (+1) or negative (-1). The goal of SVM is to find a hyperplane in the feature space that maximally separates the two classes.
Mathematically, we can represent the hyperplane as a linear equation:

$$w \cdot x + b = 0 \tag{5.5}$$

where w is the normal vector to the hyperplane and b is the bias term. The sign of $w \cdot x + b$ determines on which side of the hyperplane a data point lies. If $w \cdot x + b > 0$, then the data point belongs to the positive class, otherwise it belongs to the negative class.

To find the optimal hyperplane, SVM aims to maximize the margin between the hyperplane and the closest data points from each class. These data points are known as support vectors, hence the name "Support Vector Machines". The margin is defined as the perpendicular distance between the hyperplane and these support vectors.

Let's denote the set of support vectors as S. For any given data point xi in S, we have:

$$w \cdot x_i + b = 1 \text{ if } y_i = +1 \tag{5.6}$$
$$w \cdot x_i + b = -1 \text{ if } y_i = -1 \tag{5.7}$$

where yi represents the class label of xi. The margin can be calculated as:

$$\text{margin} = \frac{w}{\|w\|} \cdot (x_i^+ - x_i^-) = \frac{2}{\|w\|} \tag{5.8}$$

where x_i^+ and x_i^- are two support vectors from the positive and negative classes, respectively. The objective of SVM is to maximize this margin, which can be formulated as an optimization problem.

To handle cases where the data is not linearly separable, SVM introduces the concept of slack variables. These variables allow for a certain amount of mis-classification or overlapping between the classes. Let's denote the slack variables as ξ_i, where $\xi_i \geq 0$ for all data points. The optimization problem can then be formulated as:

$$\begin{aligned} \text{minimize: } & \frac{1}{2}\|w\|^2 + C\sum \xi_i \\ \text{subject to: } & y_i(w \cdot x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{5.9}$$

where C is a hyperparameter that controls the trade-off between maximizing the margin and minimizing the misclassification errors. A larger value of C allows for fewer misclassifications but may result in a smaller margin, while a smaller value of C allows for a larger margin but may lead to more misclassifications.

To solve this optimization problem, we can use techniques from convex optimization, such as quadratic programming or Lagrange duality. The solution will provide us with the optimal values of w and b, which define the hyperplane that separates the classes.

5.4 UML Diagrams

5.4.1 Use Case Diagram

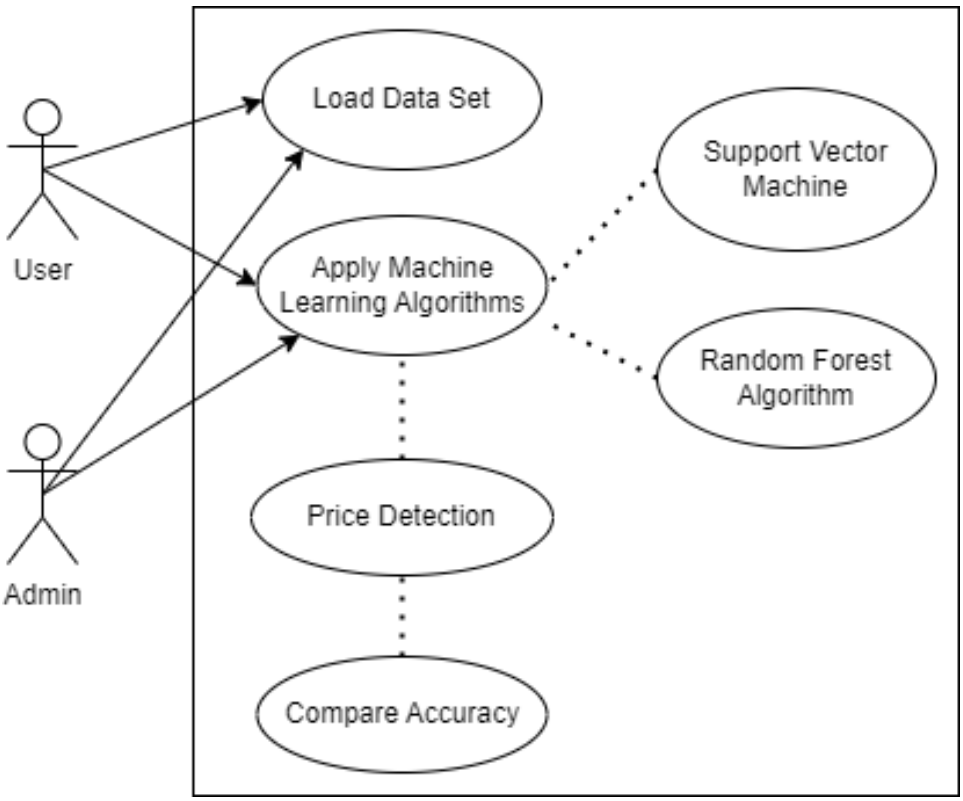

Figure 5.4.1.1: Use Case Diagram

By implementing a house price prediction model, the real estate agency can streamline its property valuation process, provide more accurate pricing recommendations, and offer better services to its clients, ultimately leading to increased customer satisfaction and business success.
Actors: User

Use Cases:

1. Configure System
2. Train Model
3. Deploy Model
4. Monitor System

Users would be able to log into the model to view the following:

1. The end use enters the relevant paramater of the house they want to price in to the model.
2. The preprocessed parameter are fed in to the trained machine learninfg model .
3. The model outputs a predicted price for the house based on the provided parameters.
4. This prediction can be displayed to the end user through the same interface they used to input the parameters..

5.4.2 Activity Diagram

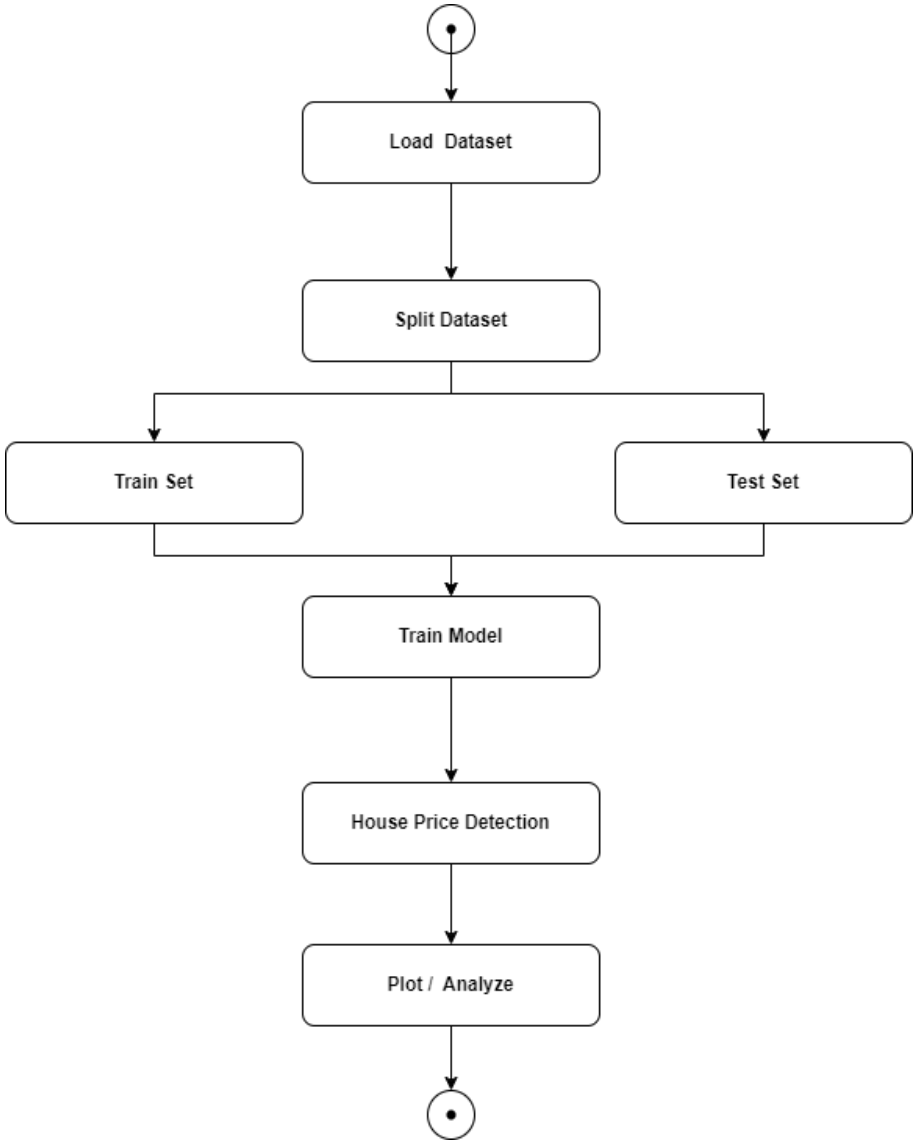

Figure 5.4.2.1: Activity Diagram

This activity diagram shows the high-level steps involved in developing and deploying an house price prediction system using machine learning.
Actors:

- **System administrator**: Responsible for developing, deploying, and monitoring the house price detection system. Short description for project report:

Activities:

1. **Collect and prepare data**: Collect network traffic data and prepare it for training and testing the machine learning models.
2. **Train machine learning models**: Train two machine learning models, SVM and random forest, on the prepared data.
3. **Evaluate machine learning models**: Evaluate the performance of the two machine learning models on a held-out test set.
4. **Monitor house price detection system**: Monitor the home price prediction system for performance and security issues.

5.4.3 Sequence Diagram

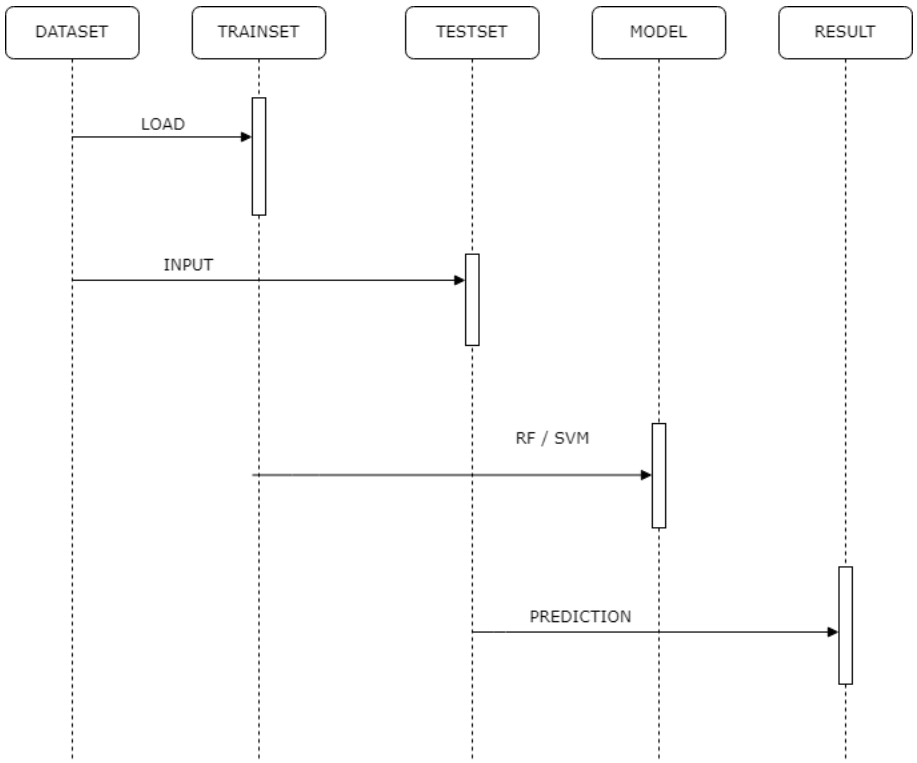

Figure 5.4.3.1: Sequence Diagram

The sequence diagram provides a high-level overview of the steps involved in the home price prediction system using machine learning. It shows the interactions between the different components of the system, including the user, and the house price detection model.

The sequence diagram shows the following steps involved in the home price prediction system using machine learning:

1. The user provides a request to the system in the form of input needed.
2. Request will be forwarded to the house price prediction model.
3. The house price detection model analyzes the request and generates a prediction.
4. The house price prediction model returns the prediction on to the system.
5. The model displays the prediction to the user.

5.4.4 Class Diagram

Class Diagram for House Price Prediction System Using Machine Learning with 3 classes:
Detects_Price: This class represents the overall house price detection system. It has the following attributes and methods:

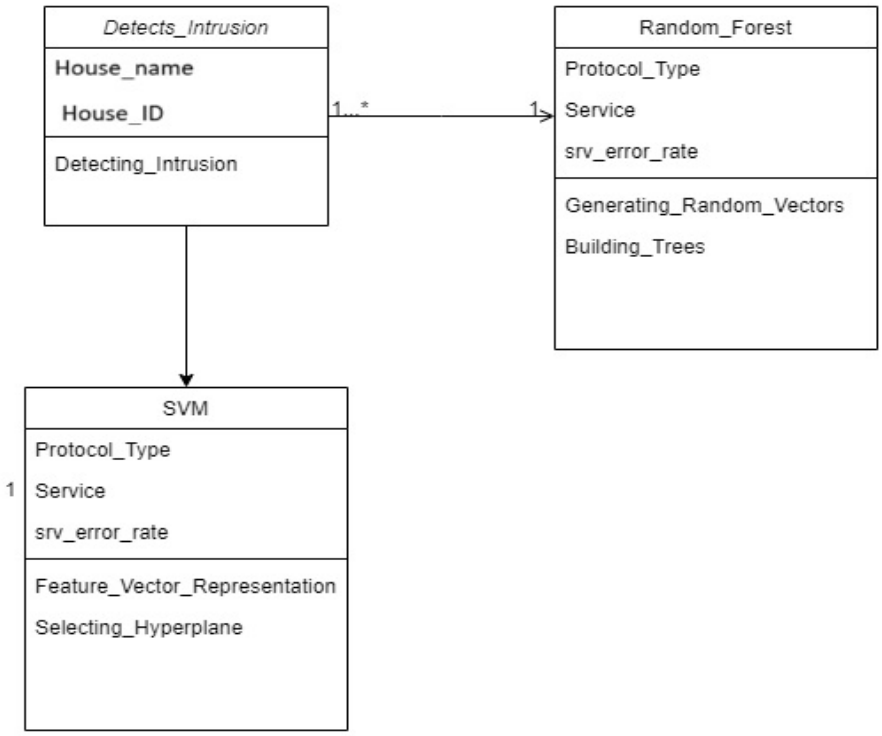

Figure 5.4.4.1: Class Diagram

- **House_ID:** A unique identifier for the intrusion.
- **House_Name:** The type of intrusion.
- **Detecting_Price( ) :** This method takes input and returns a prediction of a particular house price.

Random_Forest: This class represents a random forest machine learning model. It has the following attributes and methods:

- **protocol_type:** Type of Protocol(TCP, UDP...)
- **service:** Destination Service(ftp, telnet...)
- **srv_error_rate:** Percentage of connections with SYN errors

  and many more features.
- **Generating_random_vectors( )** This method generates random vectors from the network traffic features.
- **Building_trees( )** This method builds a set of decision trees from the random vectors.

SVM: This class represents a support vector machine (SVM) machine learning model. It has the following attributes and methods:

protocol_type: Type of Protocol(TCP, UDP...) **service:** Destination Service(ftp, telnet...) **srv_error_rate:** Percentage of connections with SYN errors
and many more features.

Chapter 6

EXPERIMENTAL SETUP

6.1 Introduction:

In response to the demand for accurate real estate predictions, this project aims to develop a robust house price prediction model. Through meticulous dataset selection, preprocessing, model training, and evaluation, our objective is to create a reliable tool capable of forecasting house prices based on key property features. By leveraging advanced regression algorithms and thorough validation processes, we seek to ensure the model's accuracy and reliability. Our focus is on empowering stakeholders, including buyers, sellers, and investors, with valuable insights derived from historical housing data.

6.2 Dataset Selection:

Choose a comprehensive dataset containing historical housing data. Ensure the dataset includes relevant features such as location, size, number of bedrooms/bathrooms, amenities, and sale prices. Consider sources like Kaggle, government databases, or real estate agencies.

6.3 Data Preprocessing:

1. Handle missing values: Impute or remove missing data using techniques like mean imputation or advanced imputation methods.
2. Encode categorical variables: Convert categorical features into numerical format using techniques like one-hot encoding or label encoding.
3. Feature scaling: Standardize or normalize numerical features to ensure all features are on a similar scale.
4. Outlier detection and removal: Identify and handle outliers that may adversely affect model performance.

6.4 Model Training:

1. Select appropriate regression algorithms such as linear regression, decision trees, random forests, or gradient boosting.
2. Split the dataset into training and testing sets (e.g., 80-20 split).
3. Train the selected models on the training data.
4. Tune hyperparameters using techniques like grid search or random search to optimize model performance.

6.5 Model Evaluation:

1. Evaluate models using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), or R-squared.
2. Compare the performance of different models on the testing set and select the best-performing one.
3. Perform cross-validation to assess model robustness and generalization.

6.6 Integration with Frontend:

If a frontend interface is required, integrate the model within the frontend using appropriate programming languages and frameworks. This could involve developing user interfaces for data input and result display.

6.7 Evaluation and Analysis Display:

Display the model's predictions and analysis results in a clear and understandable format. This could include visualizations such as charts, graphs, or summary statistics to aid interpretation.

6.8 Details about Inputs to the System

1. Property Features: Essential input includes property attributes such as size (square footage), number of bedrooms and bathrooms, location (including geographical coordinates), age of the property, and architectural style.
2. Neighborhood Characteristics: Information about the neighborhood, including crime rates, school district quality, proximity to amenities (such as parks, shopping centers, and public transportation), and overall neighborhood desirability.

3. Market Indicators: Relevant market indicators like historical property sales data, trends in housing prices over time, interest rates, inflation rates, and economic indicators affecting the real estate market.

4. External Factors: Other external factors that may influence house prices, such as demographic trends, employment rates, GDP growth, and regulatory changes impacting the housing market.

5. Additional Features: Optional features such as property condition, renovation history, presence of amenities (such as swimming pools or outdoor spaces), and any unique selling points that could affect the property's value.

6. Data Preprocessing Results: Preprocessed data, including cleaned and standardized features, handling missing values, encoding categorical variables, and any transformations applied to the input features to prepare them for model training.

6.9 Evaluation Parameters

1. **Accuracy:** The accuracy of the house price detection system is the percentage of correctly classified data points. It is calculated as follows:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

where,

- TP (true positive) is the number of correctly classified malicious data points.
- TN (true negative) is the number of correctly classified normal data points.
- FP (false positive) is the number of normal data points that are incorrectly classified as malicious.
- FN (false negative) is the number of malicious data points that are incorrectly classified as normal.

2. **Precision:** The precision of the intrusion detection system is the percentage of predicted positive cases that are actually positive. It is calculated as follows:

$$Precision = TP/(TP + FP)$$

3. **Recall:** The recall of the intrusion detection system is the percentage of actual positive cases that are correctly predicted. It is calculated as follows:

$$Recall = TP/(TP + FN)$$

4. **F1 Score:** The F1 score is a harmonic mean of precision and recall. It is calculated as follows:

$$F1score = 2 * (Precision * Recall)/(Precision + Recall)$$

5. **Area under the ROC curve (AUC):** The AUC is a measure of the overall performance of a classifier. It is calculated by plotting the true positive rate (TPR) against the false positive rate (FPR) at different thresholds. The AUC ranges from 0 to 1, with a higher AUC indicating better performance.

6. **ROC curve:** The ROC curve is a graphical representation of the performance of a classifier. It is plotted by plotting the TPR against the FPR at different thresholds. The ROC curve can be used to visualize the trade-off between sensitivity and specificity.

6.10 Software and Hardware Setup

6.10.1 Software and Packages Requirements:

1. **Machine Learning Tools:**

   (a) **Python:** Python will be used to implement the entire system, including the data preprocessing, feature extraction, machine learning model training and evaluation.

   (b) **Scikit-learn:** Scikit-learn will be used to train and evaluate the machine learning models for house price prediction.

   (c) **Jupyter Notebook or Jupyter Lab:** Useful for experimenting and documenting your machine learning code.

2. **Web Application:**

   (a) **Flask:** Flask will be used to develop a model that can be used to interact with the house price prediction system.

   (b) **HTML CSS:** HTML, CSS will be used to develop the user interface (UI) of the model that is used to interact with the house price prediction system.

3. **Data Analysis:**

   (a) **Pandas:** Pandas will be used to preprocess the house price data and extract features that are relevant for house price prediction.

   (b) **Matplotlib:** Matplotlib will be used to visualize the house price data and the results of the machine learning model evaluation.

   (c) **Numpy:** Numpy will be used for numerical operations and linear algebra computations.

4. **Databases:**

   (a) **PostgreSQL:** A powerful open-source relational database management system.

   (b) **MongoDB:** A NoSQL database for flexible data storage and retrieval.

5. **Model Deployment:**

   (a) Tools like Docker and Kubernetes for containerization

   (b) Amazon SageMaker or other cloud-based machine learning platforms for model deployment.

6.10.2 Hardware Requirements:

1. **Operating System:** These editions offer features and compatibility suitable for development and deployment

2. **Memory 16 GB RAM or more:** Machine learning tasks, especially when working with larger datasets, benefit from ample RAM. A minimum of 16 GB is recommended for smooth development and training experiences.

Chapter 7

IMPLEMENTATION PLAN OF NEXT SEMESTER

7.1 Implementation Plan:

Table 7.1: Project Implementation Plan

Week	Tasks	Description
1-2	Project kickoff, requirements gathering, and dataset preparation.	Define project scope, objectives, and deliverables. Identify specific user requirements for the web application.
3	Research and dataset preparation.	Acquire the Indian House datasets. Preprocess and clean the datasets, handling missing values and outliers.
4-5	Algorithm selection and development environment setup.	Select the machine learning algorithms (SVM and Random Forest) for price detection. Set up the development environment with Python, Jupyter Notebooks, and necessary libraries (scikit-learn, Flask, etc.).
6-7	Model training and evaluation.	Divide the dataset into training and testing sets. Train and evaluate the SVM and Random Forest models.
8-9	Web application development with Flask.	Design the user interface using HTML, CSS, and JavaScript. Integrate the trained models into the web application for real-time detection.
10	Evaluation, analysis, and user testing.	Implement evaluation metrics (e.g., accuracy, precision, recall, F1-score) for the models. Create visualizations and dashboards to display results to users.
11-13	Documentation, final adjustments, presentation, and project wrap-up.	Document the project, including code, architecture, and user instructions.

Chapter 8

CONCLUSION

House price prediction is crucial for the real estate market. By leveraging advanced modeling techniques, such as machine learning algorithms, we can make more accurate predictions. These predictions benefit buyers, sellers, investors, and policymakers by providing valuable insights into market trends and facilitating informed decision-making. Continued research in this field holds the potential to further improve prediction accuracy and enhance the efficiency of real estate transactions.

REFERENCES

[1] H. D. B. M. Quang Troung, Mint Nguyen, "House price prediction via improved machine learning technique," *International Research Journal of Engineering and Technology (IRJET)*, vol. 07, no. 04, pp. 405–408, 2020.

[2] K. K. S. Master Amit Kumar, "House price prediction using regression technique," *Cybersecurity*, vol. 02, no. 01, pp. 2–20, Dec. 2019.

[3] P. R. I. Madras], "House price prediction using machine learning," *IOP Conference Series: Materials Science and Engineering*, vol. 1099, no. 01, Mar. 2021.

[4] S. M. Nihar Bhagwat, Ankit Mohokar, "House prediction forecasting using machine learning," *International Conference on Smart Electronics & Communication (ICOSEC 2020)*, vol. 90, no. 01, pp. 149–155, Nov. 2020.

[5] F. S. e. a. Arman Sharma, H. J Syed, "A comprehensive review of house price prediction," *Sensors 2023*, vol. 10, no. 01, pp. 1–26, Jun. 2019.